MATLAB Simulation of Inverse Planar Kinematics of a Two-Link Robot Arm Using Trigonometric Functions

MECE 617 Activity

Ivan John A. Naparota
University of San Carlos – Talamban Campus
Department of Electrical and Electronics Engineering
ivanjohnnaparota66@gmail.com

***Abstract* — concerned in finding out the joint variables in terms of the end-effector position and orientation is Inverse Kinematics, Inverse Kinematics problems can either be solved with matrix manipulations or simply by trigonometric approach, this activity shows solution for Inverse Kinematics problems with the latter approach, also, MATLAB was employed to avoid manual solutions and for simulation of the result.**

Keywords – Inverse Kinematics, Trigonometric, MATLAB

I. INTRODUCTION

As known in the previous chapters, kinematics in robotics could be divided into two, Forward or Inverse Kinematics.

Calculating the position and orientation of the end-effecter to a fixed coordinate system considering joints' variable movements is defined as Forward Kinematics.

On the other hand, Inverse Kinematics is the inverse problem of finding the joint variables in terms of the end-effector position and orientation.

Furthermore, both forward and inverse kinematics could either be solved by matrix manipulations or simply through trigonometric approach, and this is what this activity is all about, solving inverse kinematics problems with the use of trigonometric functions, laws, etc.

Referring to Drawing 1, the arm consists of two movable links that move within a two dimensional plane with X and Y being the axes. All links are connected with movable joints and the second link has an axis which is perpendicular to the first one.

II. EQUATIONS AND MATLAB CODES

A. *Equations*

As shown in Drawing 1, consider Joint 1 to be driving Link 1, and Joint 2 to be driving Link 2, motion in Joint 1 will give an angle θ_1 making Link 1 move, motion in Joint 2 gives an angle θ_2 also making Link 2 move.

Link 1 and 2 are projected on the other side to form a parallelogram with angles labeled θ_1' and θ_2', as this could be another possible orientation of the links to make the end-effecter located to point (x,y).

Letting l_1 to be the length of Link 1, l_2 be the length of Link 2, D be the diagonal of the parallelogram, α be the angle between links 1 and 2, and β be the angle between a link and the diagonal, these equations could be formulated,

Note: x and y here is the point location of the end-effecter

$$D = \sqrt{x^2 + y^2}$$

By Cosine Law:

$$D^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos\alpha$$

$$\alpha = \cos^{-1}\frac{l_1^2 + l_2^2 - D^2}{2l_1 l_2}$$

$$l_2^2 = l_1^2 + D^2 - 2l_1 D\cos\beta$$

$$\beta = \cos^{-1}\frac{l_1^2 + D^2 - l_2^2}{2l_1 D}$$

Therefore,

$$\boldsymbol{\theta_1 = \tan^{-1}\frac{y}{x} - \beta}$$

$$\boldsymbol{\theta_2 = 180 - \alpha}$$

$$\boldsymbol{\theta_1' = \tan^{-1}\frac{y}{x} + \beta}$$

$$\boldsymbol{\theta_2' = -(180 - \alpha)}$$

These equations are being transferred to MATLAB

B. *MATLAB Codes*

```
clc;
lengthlink1 = 'Length of First Link: \n';
lengthlink2 = 'Length of Second Link:
\n';
posendx = 'Position of End Effecter X:
\n';
```

```
posendy = 'Position of End Effecter Y:
\n';

a = input(lengthlink1);
b = input(lengthlink2);
c = input(posendx);
d = input(posendy);

if ((c <= (a + b)) && (d <= (a + b)));

z = sqrt((c^2) + (d^2));
A = acosd((a^2 + b^2 - z^2)/(2*a*b));
B = acosd((a^2 + z^2 - b^2)/(2*a*z));
one = atan2d(d,c) - B;
two = 180 - A;
oneprime = atan2d(d,c) + B;
twoprime = -(180 - A);

disp('Theta1:');
disp(one);
disp('Theta2:');
disp(two)
disp('OR2');
disp('Theta1:');
disp(oneprime);
disp('Theta2:');
disp(twoprime);

maxaxis = a + b;
x = a*cosd(one);
y = a*sind(one);
v = a*cosd(oneprime);
w = a*sind(oneprime);
line ([0 x],[0 y],'color','r')
line ([x c],[y d],'color','g')
line ([0 c],[0 d],'color','y')
line ([0 v],[0 w],'color','b')
line ([v c],[w d])
axis([-maxaxis maxaxis -maxaxis maxaxis])
title('2-Link Planar Inverse Kinematics')
xlabel('X-Axis')
ylabel('Y-Axis')
grid on

else
    disp('****Youre Entering Invalid
Data****');
end


disp('Type [TwoLinkInverseTrigo
] in Command Window to Input again :)')
```

III. METHODOLOGY

i. Formulate equations using Trigonometric functions
ii. Make a program in MATLAB based from the formulated equations to solve the joint variables
iii. Compare the outcome of the program to the program of Forward Kinematics
iv. Evaluate the results

IV. RESULTS AND DISCUSSIONS

As shown in Figure 2, the program will ask inputs, Length of Link 1, Length of Link 2, and the point location of the end-effecter x and y. Provided with the following inputs:

Length of Link 1: 4 units

Length of Link 2: 3 units

Position of End-Effecter X: 5.7622

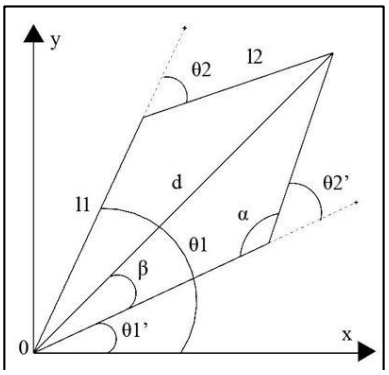
Position of End-Effecter Y: 0.0716

We'll get two sets of angles as outcomes, $\theta_1$and $\theta_1{}'$ are -28.5762° and 30°, $\theta_2$and $\theta_2{}'$ are 70° and -70°. These are the joint variables that would define the end-effecter's position and orientation, for the plot, refer to Figure 1. These results are considered to be correct if we'll refer to the result of the previous activity of Forward Kinematics when provided with above angles and the same lengths of links (*refer to Figure 3*).

V. CONCLUSIONS

The joint variables that would define the end-effecter's position and orientation in a two dimensional plane can be solved with inverse kinematics, also, trigonometric approach can be used in solving inverse kinematics problems. Finally, MATLAB is of very much help to avoid manual solutions and to be able to simulate results.

VI. DRAWINGS AND FIGURES



Drawing 1. 2 Link Planar Kinematics

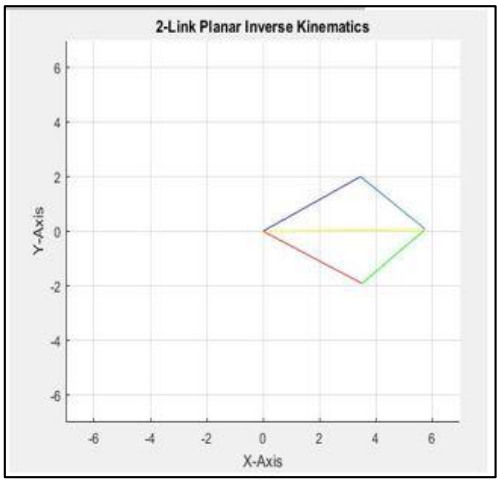


Figure 2. MATLAB plot for Inverse Kinematics

```
Length of First Link:
4
Length of Second Link:
3
Position of End Effecter X:
5.7622
Position of End Effecter Y:
0.0716
Theta1:
  -28.5762

Theta2:
   70.0010

OR2
Theta1:
   30.0000

Theta2:
  -70.0010
```

Figure 1. MATLAB Command Window

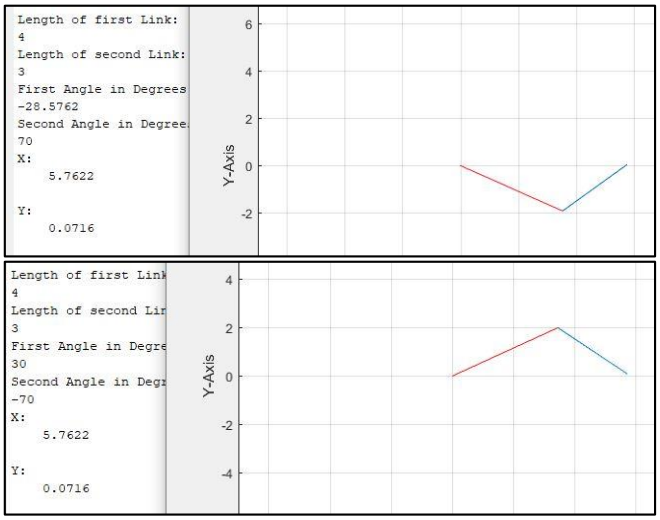


Figure 3.MATLAB Plot and Results of Forward Kinematics